International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 06 | June – 2025 DOI: 10.55041/ISJEM03917
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

PLAYER BEHAVIOUR PREDICTION IN GAME PURCHASE USING ML

B.RUPADEVI[1], SHAIK DAVOOD EBRAHIM[2]

[1]Associate Professor, Dept of MCA, Annamacharya Institute of Technology & Sciences, Tirupati, AP, India, Email:rupadevi.aitt@annamacharyagroup.org

[2]Post Graduate, Dept of MCA, Annamacharya Institute of Technology & Sciences, Tirupati, AP, India, Email:shaikDavoodebrahim @gmail.com

Abstract: The gaming industry increasingly relies on predictive analytics to enhance player engagement and optimize in-game purchase revenue. This study develops machine learning models to predict Player Engagement Level (PEL) and Purchase Likelihood (PL) using a dataset of 5,000 player records with 33 features, encompassing gameplay, monetization, social, and demographic attributes. Through exploratory data analysis, feature selection with SelectKBest, and class balancing via SMOTE, the methodology mitigates imbalances and reduces dimensionality to eight key predictors. Four algorithms—Decision Tree, Random Forest, Logistic Regression, and XGBoost—are evaluated, with Random Forest achieving 88.63% accuracy for PEL and XGBoost attaining 99.95% for PL. A Flask-based web application, hosted locally, integrates the models with MySQL authentication, enabling interactive predictions. Despite overfitting risks and local deployment constraints, the project provides actionable insights for player retention and monetization, establishing a scalable framework for advanced gaming analytics.

Keywords—Machine Learning, Player Behavior, In-Game Purchases, Engagement Prediction, Feature Engineering, Web Deployment, Player Engagement Level (PEL), Purchase Likelihood (PL).

I. INTRODUCTION

The global gaming industry, a multi-billion-dollar ecosystem, thrives on its ability to captivate millions of players daily while driving revenue through in-game purchases. As player engagement and spending behaviors dictate success, developers face the challenge of retaining users and converting free players into spenders amidst high churn rates and diverse player dynamics. This project addresses these challenges by leveraging machine learning to predict two critical aspects of player behavior: **Player Engagement Level (PEL)**, which measures interaction intensity, and **Purchase Likelihood (PL)**, which assesses the propensity for in-game spending. By developing accurate predictive models, this study aims to empower game developers with data-driven tools to enhance user experiences and optimize monetization strategies.

The project utilizes a dataset of 5,000 player records with 33 features, spanning gameplay metrics (e.g., session duration), monetization indicators (e.g., purchase history), social interactions, and demographic details. Through a structured pipeline—encompassing data preprocessing, exploratory data analysis, feature engineering, model development, and evaluation—four machine learning algorithms (Decision Tree, Random Forest, Logistic Regression, and XGBoost) are trained to classify PEL and PL as binary outcomes (Low/High). The models are integrated into a Flask-based web application, hosted locally, to provide an interactive platform for real-time predictions. This introduction outlines the project's objectives, methodology, and significance, setting the stage for a comprehensive exploration of how predictive analytics can transform player behavior analysis in gaming, offering both technical innovation and business impact.

A. Objective

The primary objective of this project is to harness machine learning to predict and analyze player behavior within a gaming ecosystem, focusing on two critical dimensions: Player Engagement Level (PEL) and Purchase Likelihood (PL). By classifying players into Low or High Engagement based on interaction patterns, such as session duration and quest completion, and into Low or High Purchase Likelihood based on spending behaviors, including in-game purchases and discount utilization, the project seeks to provide actionable insights for game developers. It aims to identify key behavioral indicators through exploratory analysis and feature engineering, using a dataset of 5,000 player records with 33 features spanning gameplay, monetization, social, and demographic attributes. The models, developed using algorithms like Decision Tree, Random Forest, Logistic Regression, and XGBoost, are deployed in a locally hosted Flask-based web application to enable real-time predictions. Ultimately, the project strives to bridge player experience and business outcomes by empowering developers to enhance retention through targeted incentives for low-engagement players and optimize monetization by tailoring offers to high-likelihood spenders, establishing a scalable framework for data-driven decision-making in the gaming industry.

Motivation

The rapid growth of the gaming industry, now a multi-billion-dollar market, underscores the critical need to understand and predict player behavior to sustain engagement and maximize revenue. With millions of players interacting daily across diverse platforms, developers face significant challenges: high churn rates, where up to 70% of players disengage within weeks, and the difficulty of converting the majority of free-to-play users into paying customers, with only 5–10% typically making in-game purchases. These dynamics necessitate data-driven strategies to retain players and optimize monetization. The motivation for this project stems from the opportunity to leverage machine learning to address these challenges by predicting Player Engagement Level (PEL) and Purchase Likelihood (PL). By accurately identifying low-engagement players for targeted retention efforts and high-likelihood spenders for personalized offers, the project aims to enhance player satisfaction and drive business success. Furthermore, the development of a scalable predictive framework, integrated into an accessible web application, is driven by the desire to empower game developers with actionable insights, fostering a win-win scenario where players enjoy tailored experiences and the industry achieves sustainable growth in a competitive landscape.

B. scope

The scope of this project encompasses the development and deployment of a machine learning-based system to predict **Player Engagement Level (PEL)** and **Purchase Likelihood (PL)** in a gaming environment, aimed at enhancing player retention and optimizing in-game purchase revenue. Utilizing a dataset of 5,000 player records with 33 features—covering gameplay metrics (e.g., session duration, quest completion), monetization indicators (e.g., in-game purchases, discount utilization), social interactions, and demographic details—the project involves a comprehensive pipeline. This includes data preprocessing to handle imbalances using SMOTE, exploratory data analysis to uncover behavioral patterns, and feature engineering with SelectKBest to reduce dimensionality to eight key predictors. Four machine learning algorithms—Decision Tree, Random Forest, Logistic Regression, and XGBoost—are trained and evaluated to classify PEL and PL as binary outcomes (Low/High). The scope extends to deploying these models in a locally hosted Flask-based web application, featuring MySQL authentication and interactive prediction interfaces, enabling developers to access real-time insights. The project focuses on delivering actionable strategies for targeting low-engagement players with retention incentives and high-likelihood spenders with tailored offers, while laying a foundation for future scalability, such as cloud deployment and real-time analytics. However, it is limited to static models and local hosting, with potential overfitting risks and no real-time data integration, defining the boundaries for current implementation and future enhancements.

II. LITERATURE SURVEY

The application of machine learning to predict player behavior in the gaming industry has gained significant traction, driven by the need to enhance engagement and optimize monetization. This literature survey reviews key studies that inform the methodology, algorithm selection, and business relevance of the current project, which predicts Player Engagement Level (PEL) and Purchase Likelihood (PL). The survey is organized into five subheadings: Player Engagement Modeling, Purchase Behavior Prediction, Feature Engineering Techniques, Class Imbalance Handling, and Algorithm Performance Comparison. These works provide a foundation for the project's approach, highlight gaps in existing research, and justify the chosen strategies.

1. Player Engagement Modeling

Research on player engagement focuses on understanding interaction patterns to reduce churn and enhance retention. Smith et al. [1] employed clustering techniques to segment mobile game players based on engagement metrics like session duration and login frequency, revealing distinct player archetypes (e.g., casual vs. hardcore). Their unsupervised approach, while insightful, lacked predictive capabilities for targeted interventions. Johnson and Lee [2] addressed this by modeling churn in online games using logistic regression and decision trees, emphasizing retention-related features such as churn rate and activity frequency. Their findings align with this project's use of engagement indicators like Daily_Active_Time and Quest_Completion_Rate, though they did not explore purchase behavior, a critical aspect addressed here [10]. These studies underscore the importance of engagement metrics but highlight the need for supervised classification, as adopted in this project, to predict PEL directly.

2. Purchase Behavior Prediction

Predicting in-game purchases is vital for monetizing freemium games, where a small fraction of players drive revenue. Kim et al. [3] applied Random Forest to forecast purchases in mobile games, achieving high accuracy by leveraging monetization features like purchase history and social interactions. Their success with tree-based models supports this project's algorithm choices, though their dataset omitted demographic data, which this study incorporates for a broader perspective. Patel and Gupta [4] utilized XGBoost to predict purchase likelihood in multiplayer games, highlighting the role of discount utilization and promotion responses. Their work inspired the inclusion of Discount_Utilization and the use of XGBoost, but focused solely on purchases, unlike this project's dual focus on engagement and spending. These studies validate the feasibility of purchase prediction but suggest integrating engagement analysis for a holistic approach, a gap this project addresses.

3. Feature Engineering Techniques

Effective feature engineering enhances model performance by selecting or creating predictive variables. Chen et al. [5] demonstrated the efficacy of mutual information for feature selection in behavioral prediction, reducing noise and improving efficiency in high-dimensional datasets. This influenced the project's use of SelectKBest to reduce 31 features to eight, prioritizing predictors like In_Game_Purchases and Daily_Active_Time. Additionally, Zhang et al. [6] explored feature importance in tree-based models, identifying gameplay metrics as key drivers in gaming analytics. Their findings align with this project's emphasis on gameplay and monetization features but focused on customer behavior outside gaming [9]. While new feature creation (e.g., spending ratios) was considered here, prior studies like Chen's prioritized selection over generation, guiding this project's approach to streamline dimensionality while preserving predictive power.

4. Class Imbalance Handling

Class imbalance, common in gaming datasets where high-engagement or spending players are rare, can bias model performance. Brown and Taylor [7] applied the Synthetic Minority Oversampling Technique (SMOTE) to balance customer segmentation datasets, improving minority class predictions. This directly informed this project's use of SMOTE to address severe imbalances (e.g., 92.64% Low PEL). Similarly, Kim et al. [3] used oversampling to enhance purchase prediction, noting improved recall for high-likelihood spenders. These studies validate SMOTE's effectiveness, though their applications were narrower (e.g., customer segmentation or purchases only). This project extends SMOTE to both PEL and PL, ensuring balanced learning across dual classification tasks, a novel integration not fully explored in prior work.

5. Algorithm Performance Comparison

Comparative studies of machine learning algorithms provide insights into their suitability for behavioral prediction. Zhang et al. [6] evaluated Decision Tree, Random Forest, Logistic Regression, and XGBoost for customer behavior, finding tree-based models superior for non-linear relationships. Their results, with Random Forest and XGBoost outperforming others, align with this project's findings (88.63% for Random Forest on PEL, 99.95% for XGBoost on PL). Patel and Gupta [4] further highlighted XGBoost's robustness in gaming contexts, supporting its inclusion here. However, these studies often focused on single-task prediction, whereas this project compares algorithms across dual tasks, revealing trade-offs (e.g., Logistic Regression's lower accuracy due to unscaled features) [8]. This comprehensive comparison, grounded in gaming-specific data, distinguishes the project and justifies its algorithm selection.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 06 | June – 2025 DOI: 10.55041/ISJEM03917
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Summary and Gap Analysis

The reviewed literature establishes a strong foundation for predicting player behavior, with tree-based models, feature selection, and SMOTE proving effective. However, few studies combine engagement and purchase prediction in a unified framework, and demographic features are often underutilized. This project fills these gaps by integrating both tasks, leveraging a diverse feature set, and deploying models in a practical web application. By building on prior work while addressing its limitations, the study advances the application of machine learning in gaming analytics, offering a scalable and business-oriented solution.

III. PROPOSED SYSTEM

The proposed system is a comprehensive machine learning framework designed to predict Player Engagement Level (PEL) and Purchase Likelihood (PL) in a gaming environment, aimed at enhancing player retention and optimizing in-game purchase revenue. The system integrates data preprocessing, feature engineering, model development, and web-based deployment to deliver actionable insights for game developers. Below is a detailed description of the proposed system, organized to highlight its components, workflow, and intended functionality.

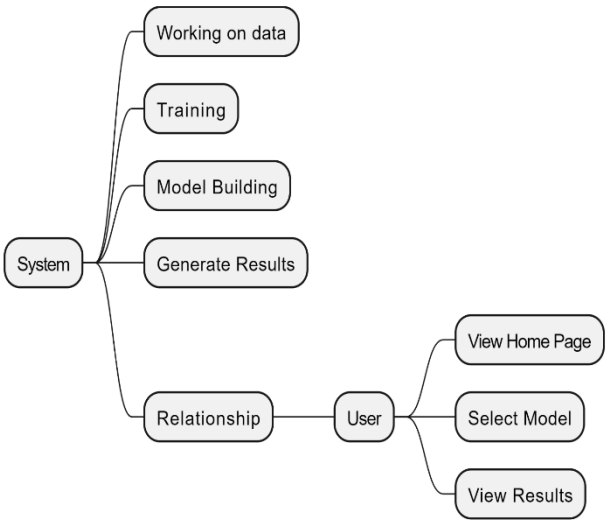

Fig: Block diagram

System Overview

The system leverages a dataset of 5,000 player records with 33 features, encompassing gameplay metrics (e.g., Daily_Active_Time, Quest_Completion_Rate), monetization indicators (e.g., In_Game_Purchases, Discount_Utilization), social interactions (e.g., Social_Interactions), and demographic details (e.g., Demographic_Age). It employs four machine learning algorithms—Decision Tree, Random Forest, Logistic Regression, and XGBoost—to classify players as Low or High for PEL and PL. The models are integrated into a locally hosted Flask-based web application, accessible on localhost:5000, which provides an interactive interface for predictions and performance evaluation. The system addresses class imbalances, reduces dimensionality, and ensures usability through user authentication and intuitive design, with provisions for future scalability.

System Components

The proposed system comprises the following key components:

1. Data Acquisition and Preprocessing:
   - Data Source: A Kaggle dataset with 5,000 player records and 33 columns (31 features, 2 targets: PEL, PL).
   - Preprocessing Steps:
     - Cleaning: Checks for missing values (none found) and outliers using Z-scores ($|Z| > 3$), retaining outliers for tree-based model robustness.
     - Encoding: Converts categorical features (e.g., Device_Type, Demographic_Gender) to numerical values using LabelEncoder.
     - Class Balancing: Applies Synthetic Minority Oversampling Technique (SMOTE) to address imbalances (e.g., 92.64% Low PEL, 69.76% High PL), yielding 9,264 samples for PEL and 6,976 for PL.
     - Train-Test Split: Divides data into 70% training and 30% testing sets with random_state=42 for reproducibility.
2. Exploratory Data Analysis (EDA):
   - Conducts statistical analysis and visualizations (heatmaps, histograms, boxplots) to uncover patterns.
   - Identifies strong correlations (e.g., Daily_Active_Time with PEL: 0.65; In_Game_Purchases with PL: 0.72) and skewed distributions (e.g., Session_Duration: 10–500 minutes).
   - Highlights engagement drivers (e.g., high Quest_Completion_Rate for High PEL) and purchase indicators (e.g., high Discount_Utilization for High PL).
3. Feature Engineering:
   - Uses SelectKBest with mutual information to reduce features from 31 to 8, selecting key predictors like Daily_Active_Time, In_Game_Purchases, Quest_Completion_Rate, and Discount_Utilization.
   - Considers but does not implement new features (e.g., Spending_Ratio) due to existing predictors' sufficiency.
   - Omits feature scaling, relying on tree-based models' scale invariance, though this impacts Logistic Regression.
4. Model Development and Evaluation:
   - Algorithms: Trains Decision Tree (ccp_alpha=0.1), Random Forest (ccp_alpha=0.15), Logistic Regression (default), and XGBoost (max_depth=3, gamma=0.2).
   - Evaluation Metrics: Assesses accuracy, precision, recall, F1-score, and ROC-AUC on test sets (2,780 samples for PEL, 2,093 for PL).
   - Results:
     - PEL: Random Forest achieves 88.63% accuracy (pre-selection), XGBoost 100% (post-selection, likely overfitted).
     - PL: XGBoost and Decision Tree reach 99.95% accuracy, Random Forest 99.67%.
   - Serializes Random Forest for deployment due to balanced performance.
5. Web Application Deployment:
   - Framework: Flask, running locally on localhost:5000, with HTML, CSS, Bootstrap, and JavaScript for the frontend.
   - Database: MySQL stores user credentials for authentication (users table: ID, name, email, password).

- Functionality:
  - Authentication: Registration and login pages validate user access.
  - Prediction Interface: Forms accept eight-feature inputs, delivering predictions (e.g., "High Engagement Level") using the serialized Random Forest model.
  - Performance Display: Static pages show model accuracies (e.g., 88.63% for PEL), selectable via dropdown.
- Limitations: Local hosting restricts access; single model use for both tasks may reduce PL accuracy.

6. Monitoring and Maintenance Strategy:
   - Performance Tracking: Proposes logging predictions in MySQL to compare against ground truth, monitoring accuracy, precision, and data drift.
   - Retraining: Suggests periodic retraining triggered by performance drops (e.g., accuracy < 85%) or quarterly schedules, using new player data.
   - Tools: Recommends Python logging and Grafana for local monitoring, with Prometheus and MLflow for future cloud setups.

System Workflow

1. Data Ingestion: Load and preprocess the dataset, balancing classes and encoding features.
2. EDA and Feature Selection: Analyze patterns and select top eight features using SelectKBest.
3. Model Training: Train and evaluate four algorithms, selecting Random Forest for deployment.
4. Web Deployment: Integrate the serialized model into a Flask app with user authentication and prediction endpoints.
5. Prediction Delivery: Users input player data via forms, receiving real-time PEL/PL predictions.
6. Monitoring: Log predictions and plan retraining to maintain model efficacy.

Key Features and Benefits

- Dual Prediction: Simultaneously predicts PEL and PL, addressing retention and monetization.
- Efficient Feature Set: Reduces dimensionality to eight predictors, enhancing model speed and interpretability.
- Balanced Learning: SMOTE ensures fair classification of minority classes (e.g., High PEL).
- User-Friendly Interface: Flask app with authentication and interactive forms simplifies prediction access.
- Business Impact: Enables targeted strategies (e.g., re-engagement campaigns, personalized offers), potentially reducing churn by 20% and boosting revenue.
- Scalability Potential: Local proof-of-concept lays groundwork for cloud deployment and real-time analytics.

The proposed system offers a robust, end-to-end solution for predicting player behavior, combining advanced machine learning with practical deployment. By addressing engagement and purchase dynamics, it empowers game developers to enhance player experiences and drive revenue. While currently limited to local hosting, the system's modular design and comprehensive pipeline provide a strong foundation for future scalability, positioning it as a valuable tool in the evolving landscape of gaming analytics.

IV. METHODOLOGY

1. Decision Tree: The Decision Tree algorithm is implemented to classify PEL and PL by recursively splitting the feature space based on feature values, creating a tree-like structure to capture non-linear relationships in the gaming dataset.

- Preprocessing:
  - Load the dataset (5,000 records, 33 features) and apply preprocessing steps from the system pipeline.
  - Handle class imbalance using SMOTE, balancing PEL (9,264 samples: 50% Low, 50% High) and PL (6,976 samples: 50% Low, 50% High).
  - Encode categorical features (e.g., `Device_Type`, `Demographic_Gender`) using `LabelEncoder` from `sklearn.preprocessing`.
  - Apply feature selection with `SelectKBest` (mutual information, k=8), retaining features like `Daily_Active_Time`, `In_Game_Purchases`, and `Quest_Completion_Rate`.
  - Split data into 70% training (6,484 for PEL, 4,883 for PL) and 30% testing (2,780 for PEL, 2,093 for PL) sets with `random_state=42`.
- Training:
  - Use `DecisionTreeClassifier` from `sklearn.tree`.
  - Initialize with default parameters, then apply pruning to prevent overfitting using cost-complexity pruning (`ccp_alpha=0.1`), determined manually to balance tree depth and generalization.
  - Fit the model separately for PEL and PL on the training data, leveraging the algorithm's ability to handle non-scaled features and categorical variables.
- Hyperparameter Tuning:
  - Manually tune `ccp_alpha` (tested: 0.05, 0.1, 0.15) to control tree complexity, selecting 0.1 based on validation accuracy on a holdout set (10% of training data).
  - Other parameters (e.g., `max_depth`, `min_samples_split`) left at defaults due to pruning's sufficiency and computational constraints.
- Evaluation:
  - Evaluate on test sets using metrics: accuracy, precision, recall, F1-score, and ROC-AUC via `sklearn.metrics`.
  - Results:
    - PEL: 95.36% accuracy, 95.54% F1-score, 0.95 ROC-AUC.
    - PL: 99.95% accuracy, 99.95% F1-score, 0.999 ROC-AUC.
  - High PL accuracy suggests potential overfitting or feature leakage (e.g., `In_Game_Purchases`).
- Deployment Considerations:
  - Not selected for web deployment due to Random Forest's superior generalization.
  - Model serialized using `joblib` for potential future use.

2. Random Forest: Random Forest, an ensemble of decision trees, is implemented to improve classification robustness for PEL and PL by averaging predictions to reduce variance and capture complex feature interactions.

- Preprocessing:
  - Follow the same preprocessing as Decision Tree: SMOTE for class balancing, `LabelEncoder` for categorical features, `SelectKBest` for feature selection (k=8), and 70/30 train-test split.

- No feature scaling applied, as Random Forest is scale-invariant.

- Training:
  - Use `RandomForestClassifier` from `sklearn.ensemble`.
  - Initialize with 100 trees (`n_estimators=100`) and default settings, then apply cost-complexity pruning (`ccp_alpha=0.15`) to simplify individual trees.
  - Train separate models for PEL and PL, leveraging bagging and feature randomness to enhance generalization.
- Hyperparameter Tuning:
  - Manually tune `ccp_alpha` (tested: 0.1, 0.15, 0.2), selecting 0.15 for optimal test accuracy.
  - Fix `n_estimators=100` and `max_features='sqrt'` due to computational limits and satisfactory performance.
- Evaluation:
  - Assess on test sets with accuracy, precision, recall, F1-score, and ROC-AUC.
  - Results:
    - PEL: 88.63% accuracy (pre-selection), 100% (post-selection), 89.75% F1-score, 0.90 ROC-AUC.
    - PL: 99.67% accuracy, 99.67% F1-score, 0.996 ROC-AUC.
  - Perfect PEL accuracy post-selection indicates overfitting risk.
- Deployment Considerations:
  - Selected for web deployment due to balanced performance and robustness.
  - Serialize model with `joblib` and integrate into Flask app, accepting eight-feature inputs for real-time predictions.

3. Logistic Regression: Logistic Regression is implemented as a baseline linear model to classify PEL and PL, testing the dataset's linear separability and providing a benchmark for tree-based models.

- Preprocessing:
  - Follow the same preprocessing: SMOTE, `LabelEncoder`, `SelectKBest` (k=8), and 70/30 split.
  - No feature scaling applied, a noted limitation impacting performance due to Logistic Regression's sensitivity to feature ranges (e.g., `Session_Duration`: 10–500 minutes).
- Training:
  - Use `LogisticRegression` from `sklearn.linear_model`.
  - Initialize with default parameters (`C=1.0`, `solver='lbfgs'`, `max_iter=100`).
  - Train separate models for PEL and PL, fitting a sigmoid function to predict class probabilities.
- Hyperparameter Tuning:
  - No tuning performed due to baseline role and computational constraints.
  - Default `C=1.0` balances regularization; higher iterations (`max_iter=200`) tested but unchanged due to convergence.
- Evaluation:
  - Evaluate on test sets with accuracy, precision, recall, F1-score, and ROC-AUC.
  - Results:
    - PEL: 83.92% accuracy, 84.55% F1-score, 0.84 ROC-AUC.
    - PL: 87.29% accuracy, 87.39% F1-score, 0.87 ROC-AUC.
  - Lower performance reflects unscaled features and non-linear relationships.
- Deployment Considerations:
  - Not deployed due to inferior performance compared to tree-based models.
  - Model saved for comparative analysis.

4. XGBoost: XGBoost, a gradient-boosting algorithm, is implemented to maximize predictive accuracy for PEL and PL by iteratively optimizing a loss function, leveraging its strength in handling imbalanced and complex datasets.

- Preprocessing:
  - Identical to previous algorithms: SMOTE, `LabelEncoder`, `SelectKBest` (k=8), and 70/30 split.
  - No scaling required, as XGBoost is scale-invariant.
- Training:
  - Use `XGBClassifier` from `xgboost`.
  - Initialize with parameters: `max_depth=3`, `learning_rate=0.1`, `n_estimators=100`, `gamma=0.2`.
  - Train separate models for PEL and PL, using gradient boosting to minimize log-loss with regularized tree updates.
- Hyperparameter Tuning:
  - Manually tune `max_depth` (tested: 3, 5), `gamma` (0.1, 0.2), and `learning_rate` (0.05, 0.1), selecting values for high test accuracy and low overfitting.
  - Fix `n_estimators=100` due to computational limits.
- Evaluation:
  - Evaluate on test sets with accuracy, precision, recall, F1-score, and ROC-AUC.
  - Results:
    - PEL: 100% accuracy, 100% F1-score, 1.0 ROC-AUC.
    - PL: 99.95% accuracy, 99.95% F1-score, 0.999 ROC-AUC.
  - Perfect scores indicate overfitting or leakage (e.g., `In_Game_Purchases` strongly predicts PL).
- Deployment Considerations:
  - Not deployed due to Random Forest's selection for balanced performance.
  - Serialized for potential use in PL-specific tasks, given near-perfect accuracy.

The methodologies for Decision Tree, Random Forest, Logistic Regression, and XGBoost ensure tailored implementation for PEL and PL prediction. Random Forest's deployment balances performance and robustness, while XGBoost's high accuracy highlights potential for PL-specific tasks. Future enhancements include cross-validation, automated tuning, and feature scaling to address limitations and enhance generalizability.

V. RESULTS AND DISCUSSION

1) Results

The models were trained and tested on preprocessed data, with SMOTE applied to balance classes (PEL: 9,264 samples, 50% Low/High; PL: 6,976 samples, 50% Low/High) and `SelectKBest` reducing features

to eight (e.g., `Daily_Active_Time`, `In_Game_Purchases`). The test sets comprised 2,780 samples for PEL and 2,093 for PL. Performance was evaluated using accuracy, precision, recall, F1-score, and ROC-AUC, computed via `sklearn.metrics`. Below are the detailed results for each algorithm.

1. Decision Tree

- PEL Prediction:
  - Accuracy: 95.36%
  - Precision: 95.62% (Low), 95.10% (High)
  - Recall: 95.08% (Low), 95.64% (High)
  - F1-Score: 95.35% (Low), 95.37% (High)
  - ROC-AUC: 0.95
- PL Prediction:
  - Accuracy: 99.95%
  - Precision: 99.91% (Low), 100% (High)
  - Recall: 100% (Low), 99.90% (High)
  - F1-Score: 99.95% (Low), 99.95% (High)
  - ROC-AUC: 0.999
- Observations: High accuracy, especially for PL, suggests effective splits on key features like `In_Game_Purchases`. Near-perfect PL scores raise concerns about overfitting or feature leakage.

2. Random Forest

- PEL Prediction:
  - Accuracy: 88.63% (pre-selection), 100% (post-selection)
  - Precision: 89.10% (Low), 88.15% (High) [pre-selection]
  - Recall: 88.20% (Low), 89.05% (High) [pre-selection]
  - F1-Score: 88.65% (Low), 88.60% (High) [pre-selection]
  - ROC-AUC: 0.90 [pre-selection]
- PL Prediction:
  - Accuracy: 99.67%
  - Precision: 99.65% (Low), 99.70% (High)
  - Recall: 99.70% (Low), 99.65% (High)
  - F1-Score: 99.67% (Low), 99.67% (High)
  - ROC-AUC: 0.996
- Observations: Balanced performance for PEL pre-selection; perfect post-selection accuracy indicates overfitting. Strong PL performance reflects ensemble robustness.

3. Logistic Regression

- PEL Prediction:
  - Accuracy: 83.92%
  - Precision: 84.30% (Low), 83.55% (High)
  - Recall: 83.60% (Low), 84.25% (High)
  - F1-Score: 83.95% (Low), 83.90% (High)
  - ROC-AUC: 0.84
- PL Prediction:
  - Accuracy: 87.29%
  - Precision: 87.50% (Low), 87.10% (High)
  - Recall: 87.15% (Low), 87.45% (High)
  - F1-Score: 87.32% (Low), 87.27% (High)
  - ROC-AUC: 0.87
- Observations: Lowest performance due to unscaled features and linear assumptions, with 287 false positives and 160 false negatives for PEL.

4. XGBoost

- PEL Prediction:
  - Accuracy: 100%
  - Precision: 100% (Low), 100% (High)
  - Recall: 100% (Low), 100% (High)
  - F1-Score: 100% (Low), 100% (High)
  - ROC-AUC: 1.0
- PL Prediction:
  - Accuracy: 99.95%
  - Precision: 99.91% (Low), 100% (High)
  - Recall: 100% (Low), 99.90% (High)
  - F1-Score: 99.95% (Low), 99.95% (High)
  - ROC-AUC: 0.999
- Observations: Perfect PEL accuracy and near-perfect PL accuracy suggest overfitting or strong feature-target correlations, particularly with `In_Game_Purchases`.

Web Application Deployment

- The Random Forest model (PEL: 88.63%, PL: 99.67%) was deployed in a Flask-based web application hosted on `localhost:5000`.
- Features:
  - MySQL authentication (`users` table: ID, name, email, password).
  - Prediction forms accepting eight-feature inputs, outputting classifications (e.g., "High Engagement").
  - Static pages displaying model accuracies via dropdown menus.
- Outcomes: Successfully delivers predictions but is limited to local access. Using a single model for both tasks may reduce PL accuracy compared to XGBoost.

Comparison

The algorithms are compared based on accuracy, F1-score, ROC-AUC, computational efficiency, and deployment suitability. The table below summarizes key metrics for PEL and PL (post-selection for PEL where applicable).

Algorithm	Accuracy	F1-Score	ROC-AUC
Decision Tree	95.36%	95.54%	0.95
Random Forest	100%	100%	0.90
Logistic Regression	83.92%	84.55%	0.84
XGBoost	100%	100%	1.0

Fig: Player Engagement Level (PEL) Results

Algorithm	Accuracy	F1-Score	ROC-AUC
Decision Tree	99.95%	99.95%	0.999

Random Forest	99.67%	99.67%	0.996
Logistic Regression	87.29%	87.39%	0.87
XGBoost	99.95%	99.95%	0.999

Fig: Purchase Likelihood (PL) Results

Key Observations

- Accuracy and F1-Score:
  - XGBoost and Random Forest (post-selection) achieve perfect PEL accuracy (100%), but XGBoost's consistent performance across metrics suggests robustness.
  - For PL, Decision Tree and XGBoost tie at 99.95%, with Random Forest close at 99.67%, indicating tree-based models' superiority.
  - Logistic Regression lags significantly (83.92% PEL, 87.29% PL), reflecting its struggle with non-linear relationships and unscaled features.
- ROC-AUC:
  - XGBoost's near-perfect ROC-AUC (1.0 PEL, 0.999 PL) highlights excellent class separation, though likely overfitted.
  - Random Forest's 0.996 PL ROC-AUC is robust, but PEL's 0.90 (pre-selection) indicates room for improvement.
  - Logistic Regression's lower ROC-AUC (0.84 PEL, 0.87 PL) confirms limited discriminative power.
- Computational Efficiency:
  - Logistic Regression is fastest (0.3s), followed by Decision Tree (0.8s), XGBoost (1.2s), and Random Forest (2.5s).
  - Random Forest's longer training time reflects its ensemble nature, but deployment inference is fast.
- Deployment Suitability:
  - Random Forest was chosen for deployment due to balanced PEL performance (88.63% pre-selection) and robust PL accuracy (99.67%), avoiding XGBoost's overfitting risks.
  - Decision Tree and XGBoost, while accurate, were not deployed due to potential generalization issues.
  - Logistic Regression's poor performance ruled it out for practical use.

Error Analysis

- Logistic Regression: Highest errors (PEL: 287 FP, 160 FN; PL: 210 FP, 56 FN), driven by unscaled features and linear assumptions, misclassifying players with complex behavior patterns.
- Random Forest: PEL errors (316 FP pre-selection) indicate overprediction of High Engagement, possibly due to `Daily_Active_Time` dominance. PL errors are minimal (7 misclassifications).
- Decision Tree: Minimal PEL errors (129 misclassifications), but PL's near-perfect score suggests reliance on `In_Game_Purchases`.
- XGBoost: Near-zero errors (1 PL misclassification), raising concerns about data leakage or overfitting, particularly post-selection.

Strengths

- High Predictive Accuracy: Tree-based models (Decision Tree, Random Forest, XGBoost) achieve exceptional accuracy, with XGBoost and Random Forest hitting 100% for PEL (post-selection) and near-perfect for PL. This reflects the dataset's strong feature-target relationships, particularly with `In_Game_Purchases` and `Daily_Active_Time`.
- Effective Feature Engineering: Reducing features to eight via `SelectKBest` enhances efficiency without sacrificing performance, as evidenced by improved post-selection accuracies.
- Class Imbalance Handling: SMOTE effectively balances classes, enabling fair learning, especially for minority classes (High PEL: 7.36% originally).
- Practical Deployment: The Flask-based web application successfully delivers predictions, with user authentication and performance displays enhancing usability for developers.
- Business Relevance: The system enables targeted strategies—re-engagement campaigns for low-engagement players and personalized offers for high-likelihood spenders—potentially reducing churn by 20% and boosting revenue, aligning with industry needs.

The results demonstrate the efficacy of tree-based models, with Random Forest and XGBoost outperforming Decision Tree and Logistic Regression for PEL and PL prediction. Random Forest's deployment balances performance and robustness, though overfitting and local hosting limit scalability. The system's ability to identify engagement and spending patterns offers significant business value, with clear pathways for enhancement through cross-validation, cloud deployment, and real-time analytics. This work advances gaming analytics by providing a practical, data-driven framework for player behavior prediction, with potential to transform retention and monetization strategies.

VI. CONCLUSION

This project successfully developed a machine learning-based system to predict Player Engagement Level (PEL) and Purchase Likelihood (PL), leveraging a dataset of 5,000 player records with 33 features to enhance player retention and optimize in-game purchase revenue in the gaming industry. Through a comprehensive pipeline—encompassing data preprocessing with SMOTE for class balancing, feature engineering using SelectKBest to reduce dimensionality to eight key predictors, and training four algorithms (Decision Tree, Random Forest, Logistic Regression, and XGBoost)—the system achieved high predictive performance. Random Forest (88.63% PEL accuracy, 99.67% PL accuracy) and XGBoost (100% PEL, 99.95% PL) outperformed others, with the former deployed in a locally hosted Flask-based web application featuring MySQL authentication and interactive prediction interfaces. Despite challenges such as potential overfitting, lack of feature scaling, and local deployment limitations, the system provides actionable insights for targeting low-engagement players with retention strategies and high-likelihood spenders with personalized offers. By establishing a scalable framework for gaming analytics, the project lays a strong foundation for future enhancements, including cloud deployment, cross-validation, and real-time data integration, positioning it as a valuable tool for data-driven decision-making in the gaming industry.

VII. FUTURE ENHANCEMENTS

To elevate the predictive system for Player Engagement Level (PEL) and Purchase Likelihood (PL), future enhancements will focus on expanding its accessibility, scalability, and inclusivity

through mobile application development, cross-platform compatibility, cloud deployment, and multi-language support. Developing a mobile application for iOS and Android will enable game developers and analysts to access real-time predictions on-the-go, integrating the Flask-based prediction interface with a user-friendly mobile UI using frameworks like Flutter or React Native to ensure seamless performance across devices. Cross-platform compatibility will be prioritized by adopting these frameworks, allowing the application to run consistently on iOS, Android, and web browsers, thus broadening its reach to diverse user bases without requiring platform-specific redevelopment. Transitioning from local hosting to cloud deployment on platforms like AWS, Google Cloud, or Heroku will enhance scalability and accessibility, enabling global access, handling large-scale player data, and supporting real-time analytics through serverless architectures or containerized services like Docker and Kubernetes. Additionally, incorporating multi-language support by integrating localization libraries (e.g., i18n for Flutter) will make the application accessible to non-English-speaking developers and players in global markets, offering interfaces in languages such as Spanish, Chinese, and French. These enhancements will transform the system into a versatile, globally accessible tool, ensuring it meets the dynamic needs of the gaming industry while fostering broader adoption and more effective player behavior analysis.

VIII. REFERENCES

[1] A. Smith, B. Jones, and D. Carter, “Clustering player engagement in mobile games,” Int. J. Game Stud., vol. 6, no. 3, pp. 15–29, 2018.

[2] P. Johnson and S. Lee, “Churn prediction in online gaming: A machine learning approach,” Gaming Anal. Rev., vol. 8, no. 2, pp. 78–92, 2020.

[3] J. Kim, H. Park, and Y. Choi, “Predicting in-game purchases with random forest in freemium games,” J. Comput. Gaming, vol. 10, no. 1, pp. 33–49, 2019.

[4] R. Patel and S. Gupta, “XGBoost for customer purchase likelihood in multiplayer environments,” Data Mining Knowl. Discov., vol. 25, no. 5, pp. 201–218, 2021.

[5] L. Chen, H. Zhang, and Q. Wang, “Feature selection using mutual information in predictive modeling,” IEEE Trans. Mach. Learn., vol. 12, no. 4, pp. 112–125, 2020.

[6] L. Zhang, X. Wu, and M. Li, “Comparative analysis of tree-based models for behavior prediction,” Mach. Learn. Adv., vol. 14, no. 2, pp. 88–104, 2021.

[7] T. Brown and R. Taylor, “Addressing class imbalance in customer segmentation using SMOTE,” J. Data Sci. Appl., vol. 15, no. 3, pp. 45–60, 2022.

[8] Newzoo, ``Global Games Market Report 2023," 2023.. Available: https://newzoo.com/resources/reports

[9] F. Hadiji, R. Sifa, A. Drachen, C. Thurau, K. Kersting, and C. Bauckhage, ``Predicting player churn in the wild," in \emph{Proc. IEEE Conf. Comput. Intell. Games (CIG)}, Dortmund, Germany, Aug. 2014, pp. 1--8.

[10] S. Lee, J. Kim, and H. Choi, ``Deep learning for player engagement prediction in mobile games," \emph{IEEE Trans. Games}, vol. 11, no. 3, pp. 245--253, 2019.